香港中文大學
The Chinese University of Hong Kong

Presented by

Jieming Zhu
jmzhu@cse.cuhk.edu.hk

Logging Practice Study: An Overview

Jieming Zhu
Oct. 15, 2013

Outline

☐ **Background**

☐ **Related Work**

☐ **Logging Practices In Industry**

☐ **Logging Suggestion**

☐ **Conclusion**

Background

- **Importance of logs**
  - Production failures are hard to reproduce
    - Privacy concerns for input
    - Hard to recreate the production setting
  - Logs are the only data available for troubleshooting failures
- **The use of logs**
  - Commercial acceptance
    - Vendors actively collect logs: Microsoft, VMware, EMC, etc.
  - Various log analysis tasks
    - Anomaly detection, error debugging, performance diagnosing, security monitoring, workload modeling, etc.

Background

What is logging?

- A log function
- Static text messages
- Optional variable values
- Verbosity levels
  - E.g., fatal/error/info/debug

Log (level, "logging message %s", variable);

```
elog (FATAL, "out of memory");                              /*PostgreSQL*/
ap_log_error (ERR, "could not open charset \        /*  Apache  */
              conversion config file %s", confname);
logit (INFO, "Authentication refused: %s", line);        /*  OpenSSH  */
elog(DEBUG1, "archived transaction log file %s", xlog); /*PostgreSQL*/
```

Logging utilities/frameworks

- General-purpose function: e.g., printf
- Logging interface: log4j (Apache), syslog (Unix)
- Logging frameworks: Event Tracing for Windows (ETW)

☐ **Hardness of logging**

- Logging too little
  - Miss valuable runtime information for postmortem analysis
  - Increase the difficulty for debugging

User:
"Apache httpd cannot start.
No log message printed."

- Logging too much
  - Runtime overhead: e.g., CPU consumption, I/O operations
  - Storage overhead: e.g., 2G of logs per machine per day
  - Produce a lot of trivial logs: redundant/useless
  - Incur additional cost of code development and maintenance

- **Optimal logging**
  - Effective logging
    - High coverage of useful information needed for postmortem analysis
  - Efficient logging
    - Low overhead and cost
    - Not producing much trivial information
- **Research problem**
  - Where to log?
    - Where to add logging statement?
  - What to log?
    - Which variables to record

Related Work

- **Ding Yuan**
  - AP, University of Toronto
  - Phd, 2012. Advisor: Yuanyuan Zhou

- **Thesis: Improving failure diagnosis via better design and analysis of log messages**
  - SherLog [ASPLOS'10]
  - LogEnhancer [ASPLOS'11]
  - ErrLog [OSDI'12]
  - Characteristic study [ICSE'12]

Yuanyuan Zhou

- Professor, UCSD

- Opera: Operating Systems Research on Energy, Reliability and Autonomy

□ **Publications**

- □ 2013
  - □ SOSP, NSDI, MiddleWare
- □ 2012
  - □ OSDI, ICSE
- □ 2011
  - □ SOSP, MICRO, FSE, ICSE, ASPLOS
- □ 2010
  - □ OSDI, OOPSLA, ASPLOS
- □ 2009
  - □ SOSP, EuroSys, ICSE, ASPLOS(2), FAST
- □ 2008
  - □ DSN(2), ASPLOS(2), ICDCS, FAST
- □ 2007
  - □ SOSP(3), FSE, EuroSys, SIGMETRICS, HPCA

☐ **How to find an idea?**

☐ **Elegant idea: Simple and effective!**

SherLog [ASPLOS'10]

- SherLog: Error Diagnosis by Connecting Clues from Run-time Logs

- Subtasks of debugging

  - Infer the failure-inducing execution path

  - Infer the conditions along the failure-inducing execution path

- **Idea:** Manual inspection → Automatic inference

- A tool:

  - Inputs: run-time logs and source code

  - Outputs: all possible failure-inducing execution path; the evolution of value on certain variables
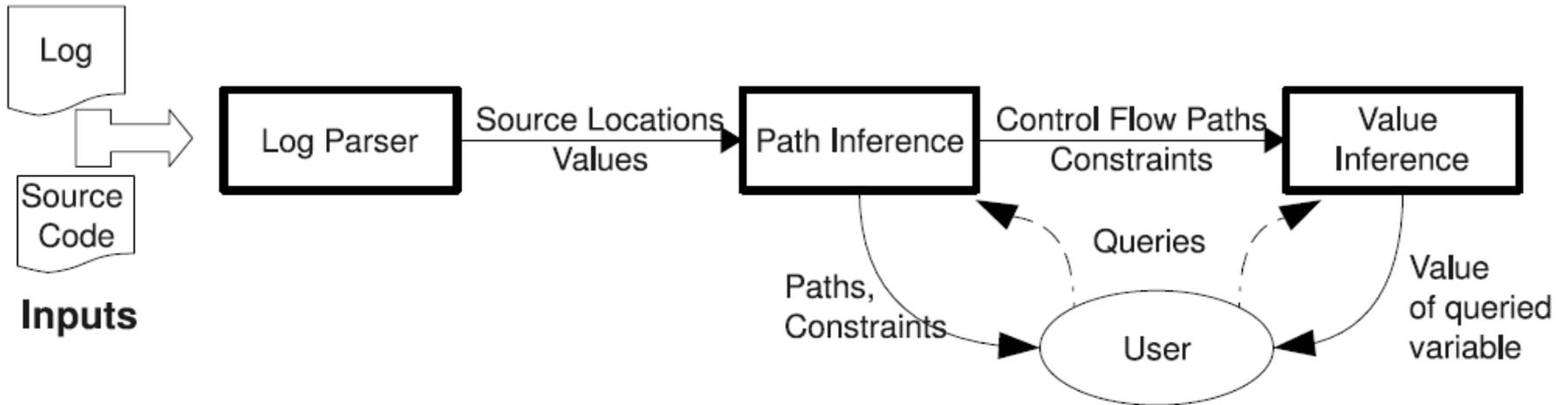
□ SherLog [ASPLOS'10]

- ◘ Log parsing
- ◘ Path Inference
- ◘ Value Inference

- **LogEnhancer [ASPLOS'11]**

  - Improving Software Diagnosability via Log Enhancement

- Idea:

  - SherLog works bad (so many candidate paths) with poor logs

  - Enhance the existing logging statements with more variable values, reducing the uncertainty in path inference

- Challenges:

  - Uncertainty identification

  - Variable selection

□ **ErrLog [OSDI'12]**

　□ Be Conservative: Enhancing Failure Diagnosis with Proactive Logging

□ Idea:

　□ LogEnhancer only works for situations where logging statements exist

　□ However, many failures have no logs at all

　□ Proactive logging: logging automatically before release

ErrLog [OSDI'12]

What are other obvious places to log?

Classic Fault-Error-Failure model [Laprie.95]

1. Function return error (30%)
2. Switch stmt. fall-through to 'default' (14%)
3. Exception signals (13%)
4. Failed input validity check (9%)
5. Resource leak (4%)
6. Abnormal exit (4%)
7. Failed memory safety check (3%)

Related Work
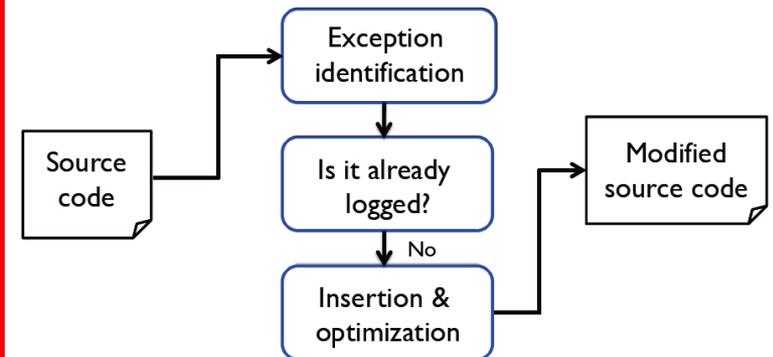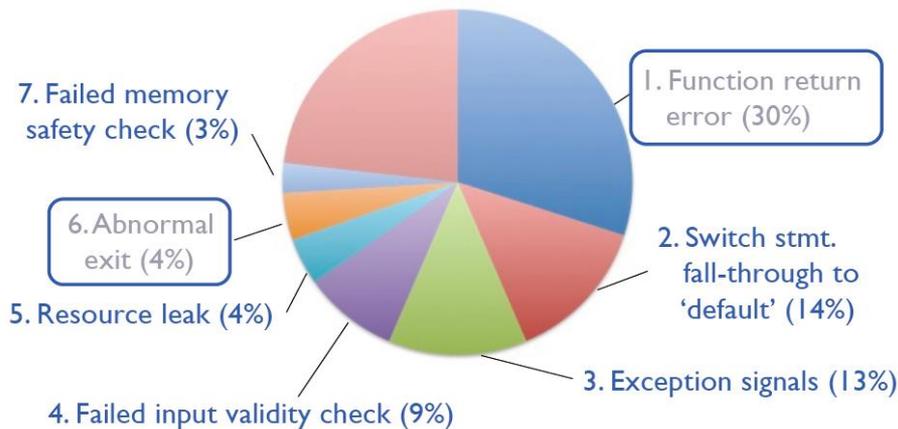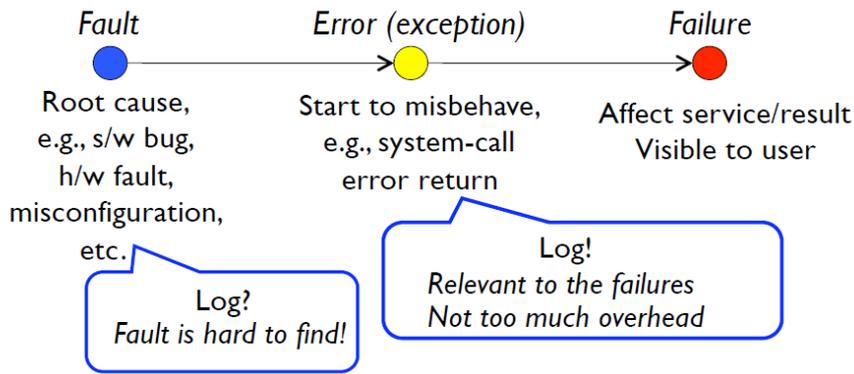
- **A characteristic study [ICSE'12]**

  - Characterizing Logging Practices in Open-Source Software

- Mining the revision histories of four open-source projects

  - Verbosity levels modification

  - Modifications to variable logging

  - Modifications to static content

  - Logging location changes

Our Work:
Logging Practices In Industry

Thank you!